REMARKS

The Applicants have amended claims 12, 18, 21, 22, 24, 27, 30, 37 and 43. The paragraph bridging pages 32 and 33 is also amended to correct what was an apparent typographic or similar error on applicant's part.

The applicant is grateful to have a review of the section 112 errors and this opportunity to correct them provided by this Office Action. While most of these errors were appropriately corrected, the applicant disagrees with the paragraph 8 objection to the lack of antecedent basis for the limitation to "claiming acts" in claim 24. Note that in claim 24, on original line 8 (counted line 2), the phrase "acts of:" and in lines 9 and 14 (counted lines 3 and 8, respectively), "claiming...". The applicant respectfully argues that taken together these words in claim 24 (read in from the independent claim 18) clearly form a sufficient antecedent basis for the phrase "claiming acts". With regard to the paragraph 10 objection to "said determinations" from claim 44, the applicant did see this phrase in claim 43 (but not in claim 44) and has modified claim 43 responsively.

The applicant respectfully traverses the rejection of claims 1-45 (as amended) on the basis of the Bertoni and Gallop references, and requests reconsideration and withdrawal of the rejection in light of the following discussion.

Bertoni is directed to creation of a flexible lock/unlock system that accommodates unique ranges of resource address, by setting up a data structure for tacking and searching for them. Bertoni does not address one of the problems highlighted in the background section of the application, that is, it is specifically designed in a way that may exacerbate a problem the applicant's invention is designed to avoid.

"A problem that arises when a resource is accessible to several threads that execute concurrently is that unpredictable, and sometimes disastrous, results can be produced if one thread is permitted to read from a resource while another thread is writing to that same resource. If reading and writing threads have simultaneous access to a resource, then the writer would be able to change the contents of the resource while the reader is in the process of reading the contents. For example, a resource could be a file of English text, and a program having a buffer that is only

10

half the size of the text reads the file in two stages. If, between these two stages, a thread reading the file is interrupted by a writing thread that changes the contents of the file, then the portion read in the second stage may no longer correspond to the portion read in the first stage, resulting in the program receiving unreadable gibberish. As another example, a user-defined data structure could represent high-precision floating-point numbers, where the mantissa is stored in the first two words of the structure, and the exponent in the last two words. A software routine that performs an arithmetic operation (e.g., multiplication) on these numbers may read the mantissa and the exponent separately. If a thread reading the number is interrupted by a writing thread between the times that the mantissa and exponent are read, then the arithmetic operation will be performed on a number fabricated from the mantissa of the old value and the exponent of the new value. In all likelihood, this fabricated number will not represent any useful value." (paragraph bridging pages 1 and 2 of the application).

Thus, Bertoni does not (as we do in claim 1) "receive(ing), from a first thread, a request for a lock (for "a resource"- from the preamble), instead Bertoni obtains information about attributes of a sub-lock, including whether a wanted flag is set (Col 5, lines 58Col 6, line 15), and the begin and end points, as well as Queue and next sub-lock information (Col 5, lines 19-25). Bertoni DECIDES whether to SPLIT THE RESOURCE, using this data. Thus, as described in Col 3 line 63 – Col 4 line8, this is something clearly not available nor wanted in the applicant's system. Further, Bertoni NEEDS to keep track of begin and end ranges and therefore cannot simply automatically destroy a sub-lock when it's not being used any more. Thus, Bertoni cannot be read on the applicant's invention since it cannot handle the automatic unlocking features present in the C++ language that the applicant' s invention uses in the preferred embodiments. Bertoni therefore is directed to providing an ability to handle splitting a resource, while the applicant's claims are directed to managing multiple thread use of a (single) "resource shared among (the) concurrently-executing threads." (Claim 1 lines 1 and 2.) Its application as a section 102 reference is therefore inappropriate as it does not provide the same functionality, does not have the same features, and does not operate in the manner of the claimed method of claim 1. Restating the missing elements of the method claim, the Bertoni reference does

not receive requests from threads, and it does not look at the desire for a resource but at a desire <u>for parts</u> of one or multiple resources. It does not check to see if another thread is reading or writing "said resource" but whether a desire is <u>for a part of a resource</u> being read from or written to. Thus, Bertoni steps into the problem stated above in applicant's page 2-3 quote above; i.e., if a portion of a resource is being written and another process seeks to read the whole resource, Bertoni may have already let the first process write while giving some kind of access to a second process for the other part of the resource that is not currently being written.

As per all of the other claims, the rejection is based on a combination of references with no clear suggestion to combine in either of said references. Particularly, with respect to claim 2 (and its dependencies), the Patent Office statement is that the combination is obvious

"... since the combination of these references provides a sound means of locking a resource. By using the framework of Gallop, a data structure is defined that allows encapsulation of threads and incorporates the locking and unlocking functions into that data structure. To limit this framework, the method of Bertoni ensures that no thread is reading or writing to the resource while another thread is writing to the resource. In doing so, data can be assured to be accurate, whereas if a thread was reading from the resource at the same time another was writing, the data may be compromised and could not be considered accurate."

But the Gallop data structure is not at all similar to the Bertoni structure, so where is the suggestion to combine? Why would one think to use Bertoni to lock the structure of Gallop? There is no actual suggestion in either case that either one would be applicable to the problem the applicant is solving, and Bertoni itself says that it may not have complete locks. "The approach (Bertoni's) also <u>allows some tradeoffs in locking accuracy</u> versus overhead and space, but in general describes the locks on a resource as a set of sub-locks." (Col 3, lines 63-65). Thus Bertoni is not directed to protection of "a resource" per se, but to ever-changing subsets of that resource. Bertoni should not be used to provide a "sound means" of locking a resource. Gallop is directed to locking objects, not parts of the objects. With respect, the applicant sees no suggestion to combine, and no operative combination either.

12

Analyzing the discussion of claim 2, it can not be seen how the Gallop reference uses a constructor to issue the request for the resource. Further, the Gallop reference specifically has two function pointers, lock and unlock, as the Patent Office points out. The applicant's claim 2 does not do this. It does not create a lock and unlock function pointer. In fact, the specification specifically mentions that unlocking is not a desirable thing to add, rather an automatic reliance on the function in C++ is used instead. "Additionally, one embodiment of the disclosed locking/unlocking mechanism permits the programmer to issue a "lock" instruction without needing to issue a corresponding "unlock" instruction." (last sentence of first paragraph of the Summary). "Since the unlocking method is called when the scope is closed by means built-in to the C++ language, it is unnecessary for the programmer to issue a separate unlocking instruction for each lock, thus minimizing the chance of programmer error. (Penultimate sentence of the Summary).

The Patent Office argument for finding Claim 4 in Gallop and Bertoni is likewise dependent on the unlock instruction, but it also suggests that the act of destroying a local class instance is obvious without any support. If Bertoni were to destroy a putative local class instance keeping track of requests to use a resource, the other related partial resource records would be lost. Accordingly, combining these references to accomplish this function is impossible.

The other dependent claims are rejected but since independent claim 1 should be allowable, addressing the individual arguments would not be productive, but the applicant does not thereby concede such argument nor admit to any statements regarding the general state of the art made in such argument by the Patent Office. Likewise, other rejections of base claims that are traversed below should apply to their dependent claims which should be allowable since the base claims should be allowable. The applicant does not concede any argument regarding dependent claims but relies on the traversal of the base claims' rejection for allowability for the dependent claims to expidite the process of consideration of these arguments.

The Patent Office argues that claim 12 is obvious from Gallop and Java as applied by a programmer. It is not believed to be within the purview of the Patent Office to apply the inventive abilities of a putative programmer to a partial recitation of elements to supply the missing elements and find a claim invalid from such an exercise. Instead, the Examiner

13

may take notice of facts, but where they are challenged, the Patent Office must supply support. MPEP 707.05 Citation of References

37 CFR 1.104 Nature of examination.
*****
(d) *Citation of references.*
(1) ...
(2) When a rejection in an application is based on facts within the personal knowledge of an employee of the Office, the data shall be as specific as possible, and the reference must be supported, when called for by the applicant, by the affidavit of such employee, and such affidavit shall be subject to contradiction or explanation by the affidavits of the applicant and other persons.

The applicant therefore respectfully requests that the basis for noticing a) that "other methods as needed" could be supplied by a Java programmer to perform as the system of the claims and b) that using lock and unlock functions within the constructor and destructor would allow for cleaner code management.

Further, in the applicant's claim 12, neither the record, object, its destructor or constructor appears to be part of the thread that is accessing or trying to access the resource, so the reference to how this rejection meets the elements of the claim is not understood since Gallop associates these lock and unlock functions with each thread, but the claim

Claim 18 is also argued to be unallowable, again by combining Bertoni and Gallop references, but Bertoni is applicable to managing sub-locks on resource sub-segments, not the overall resource. The amendment of the claim hereby makes application of these references yet less relevant as read and write critical sections are handled differently in the claim.

The next independent claim is claim 30 which is rejected at least partly because it is assumed that JAVA provides opening a local scope. JAVA also provides for opening "global" (namespace) and "class" scope, so it is believed to be incumbent upon the Patent Office to support the rejection by identifying the suggestion in the art to choose this particular scope for this particular function. Further, with the clarifying amendment to claim 30, it should be more clear that only under certain conditions will the scope be closed and the instance destroyed. Gallop does not call for destruction of any instance and again the applicant respectfully requests that the Patent Office show where in JAVA there is a call to destroy an object or even close its scope on the occurrence of such conditions.
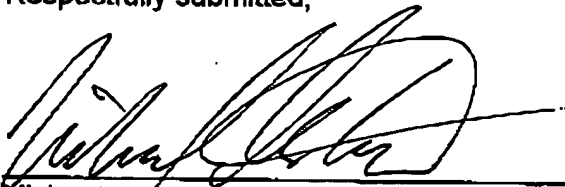
14

Claim 37, the final independent claim, is also rejected. Again the rejection is based on the Gallop reference and saying that Java could do what the applicant claims. However, it is clear that there is a system in the nature of an algorithmic process being claimed here and the applicant believes that in order to validly reject his claims, the Patent Office must show that the elements of the claim <u>are</u> met by the references, not that they <u>could be</u> met by the references. Accordingly, again, the applicant respectfully challenges the Patent Office to show actual instances of Java code having the features claimed instead of relying on supposition that it could be built with Java. There is nothing in nature that cannot be built from something else, but the inquiry that must be satisfied is that the elements claimed be found in the prior art and its suggestions. With respect, the applicant does not believe that burden has been met here and again, with respect requests reconsideration and withdrawal of the present rejection.

Believing the claims to be in condition for allowance, the applicant respectfully requests the final rejection be withdrawn and that this case be passed to issue.

Respectfully submitted,

_____   June 4, 2003
Michael B. Atlass
Attorney for Applicants
Reg. No. 30,608
Tele No. 215-986-4111

Unisys Corporation
M.S. E8-114
One Unisys Way
Blue Bell, PA 19424

/